9

Laboratorio 3
Modos de direccionamiento en el DSP

En esta sesión de laboratorio se realiza una práctica con los modos de direccionamiento y el juego de instrucciones del DSP56002. Los ejercicios permiten familiarizarse con la sintaxis del lenguaje ensamblador y aquellas instrucciones propias de un DSP. Esta práctica de laboratorio se puede realizar en el módulo de evaluación DSP56000EVM, el módulo de evaluación DSP56303EVM, el simulador GUI56000/SIM56000 o bien utilizando el simulador GUI56300/SIM56300.

Los ejercicios están basados en las notas del curso según el documento MTT31, con permiso de Freescale Semiconductors Inc. Es importante notar que los ejercicios de esta práctica son aplicables al procesador de señales DSP56303, por lo que se hará referencia como DSP56002/DSP56303.

9.1. Modos de direccionamiento

Para entender los modos de direccionamiento del DSP56002/DSP56303 es necesario conocer algunas unidades y registros internos del procesador, las cuales están encargadas de realizar las operaciones que permiten modos de direccionamiento únicos y diseñados expresamente para los algoritmos de procesamiento digital de señales.

El DSP56002 contiene las siguientes unidades:

- *Address Generation Unit* (AGU) Unidad de generación de direcciones. La AGU tiene los siguientes elementos más importantes:
  - 24 registros de 16 bits,

  - 2 ALUs de direcciones (*Aritmetic and Logic Unit* (ALU)) Unidade Aritmética y Lógica, y
  - 3 multicanalizadores de direcciones de salida.

- Registros de direcciones, los cuales están organizados en tres juegos de ocho registros:

  - Registros de direcciones `Rn n = 0,1,..,7`
  - Registros de compensación (*Offset*) `Nn n=0,1,...,7`
  - Registros modificadores `Mn  n=0,1,...7`.

Cada `Rn` tiene asociado un registro `Nn` y un registro `Mn`, los tres deben tener el mismo número `n`. `Rn` se emplea como apuntador para localizar datos en memoria. Estos datos podrán ser utilizados como operandos en algunas instrucciones. Los registros `Nn` se usan para actualizar los registros de direcciones `Rn` con algún valor de compensación. Los registros `Mn` se emplean para seleccionar el tipo de aritmética a realizar cuando se actualizan los registros de direcciones `Rn`.

Los modos de direccionamiento se muestran con algunos ejemplos. Para comenzar a trabajar, arranque el simulador GUI56000/SIM56000, o el simulador GUI56300/SIM56300.

En cada uno de los siguientes ejemplos, si es necesario, cargue el registro o los registros con el valor indicado, ensamble la instrucción que se muestra en la dirección `P` señalada y ejecute por pasos usando `STEP`. Para cargar un valor pre-establecido en algún registro utilice el mando `CHANGE`. Note que el `pc` (contador de programa) debe contener la dirección donde se ensambló la instrucción antes de intentar ejecutarla.

Ejercicio 9.1 Modo de direccionamiento directo a un registro (*Register Direct addressing mode***)**
Cargue `x1 = $332211`, y `a0=$000000`
Ensamble en el espacio de memoria P la siguiente instrucción:

```
asm p:$100  move x1,a0
```

Cambie el `pc` a $100 y ejecute un `step`.

9.1.1. Modos de direccionamiento especiales

Los modos de direccionamiento especiales indican el operando o la dirección del operando en un campo de la instrucción, o implicitamente hacen referencia al operando. Los modos de direccionamiento especiales son: dato inmediato, dato inmediato corto, dirección absoluta, dirección absoluta corta, dirección de I/O corta, salto corto a dirección, e implícito. En la tabla 9.1.1 se muestran los modos de direccionamiento.

Tabla 9.1: Modos de direccionamiento.

Modo de direccionamiento	Usa `Mn`	Usa `Rn`	Sintaxis
Directo a Registro			
Dato o Reg de control	No	-	
Registro `Rn`	No	Sí	`Rn`
Registro `Nn`	No	No	`Nn`
Registro `Mn`	No	No	`Mn`
Reg. de dir. indirecto			
Sin actualizar	Sí	Sí	`(Rn)`
Postincremento por 1	Sí	Sí	`(Rn)+`
Postdecremento por 1	Sí	Sí	`(Rn)-`
Postincremento por `Nn`	Sí	Sí	`(Rn)+Nn`
Postdecremento por `Nn`	Sí	Sí	`((Rn)-Nn`
Indexado por `Nn`	Sí	Sí	`(Rn+Nn)`
Predecremento por 1	Sí	Sí	`-(Rn)`
Especial			
Dato inmediato (24 bits)	No	No	#xxxxxx
Dirección absoluta (16 bits)	No	No	xxxx
Dato corto inmediato (8 bits)	No	No	#xx
Salto corto a dirección (12 bits)	No	No	xxx
Dirección corta absoluta (6 bits+ceros)	No	No	aa
Dirección E/S corta (6 bits+unos)	No	No	pp
Implícito	No	No	

En cada uno de los siguientes ejemplos, si es necesario, cargue el registro o los registros con el valor indicado, ensamble la instrucción que se muestra en la dirección `P` señalada y ejecute por pasos usando `STEP`. Para cargar un valor pre-establecido en algún registro utilice el mando `CHANGE`.

Ejercicio 9.2 Transferencia inmediata de un dato a un acumulador de 24
Sea `a=$0`
```
asm p:$201 move#$112233,a0
```

Ejercicio 9.3 Transferencia de un dato modo inmediato a un acumulador
```
asm p:$203 move #$112233,a
```

Ejercicio 9.4 Transferencia de un dato negativo modo inmediato a un acumulador
```
asm p:$205 move #$801111,a
```

Ejercicio 9.5 Transferencia de un dato corto modo inmediato a un acumulador a0,o a1
```
asm p:$207 move #$FF,a1
```
Si el destino es `a2, a1, a0, b2, b1, b0, R0-R7`, o `N0` a `N7`, el operando es tomado como entero sin signo.

Ejercicio 9.6 Transferencia de un dato corto modo inmediato a un registro
```
asm p:$203 move #$FF,y0
```
El operando es tomado como una fracción con signo.

Ejercicio 9.7 Transferencia de un dato corto modo inmediato a un acumulador
```
asm p:$203 move #$3F,b
```

Ejercicio 9.8 Transferencia de un dato corto negativo modo inmediato a un acumulador
```
asm p:$203 move #$83,a
```
Compare los resultados en los acumuladores `a` y `b`.

Ejercicio 9.9 Transferencia de un dato largo modo absoluto
Cargue la dirección de memoria $1000 del espacio de memoria Y con el dato $444444. Ensamble y ejecute
```
asm p:$203 move $Y:$1000,a0
```
Revise el contenido de `a0` y de Y:$1000

Ejercicio 9.10 Dato corto modo absoluto (6 bits extendido con ceros)
Cargue `a1` con $334455
```
asm p:$203 move a1,X:3
```
Observe el resultado en la memoria X con dirección $0003. Este modo trabaja en las 64 palabras bajas de memoria, es decir, de la dirección $0000 a $0064

Ejercicio 9.11 I/O corto (6 bits extendido con unos)
Cargue a1 con $123456
```
asm p:$203  move a1,X:<<$FFFE
```

La dirección X:$FFFE pertenece al espacio de periféricos incluidos en el DSP, y corresponde al registro de control del canal (bus) B. Este espacio va de la dirección $FFC0 a $FFFF. Después de ejecutar la instrucción vea el contenido de X:$FFFE.

9.1.2. Modos de direccionamiento indirecto con registro de direcciones

En este modo la instrucción utiliza un registro `Rn` para apuntar hacia un operando en memoria. Se tienen 7 posibilidades, según se muestra en los siguientes ejemplos.

Ejercicio 9.12 Sin actualización
Cargue `R0` con $2000 y `a1` con $55555. Ensamble y ejecute
```
asm p:$200 move a1,X:(R0)
```
Anote sus observaciones.
Nota `M0`=$FFFF.

Ejercicio 9.13 Sin actualización
Cargue `R0` con $2000 y `a1` con $55555. Ensamble y ejecute
```
asm p:$200 move a1,X:(R0)
```
¿Cuál es el valor de `R0`?
¿Cuál es el contenido de la dirección $2000 en la memoria X?
¿ Cuál es el contenido de `a1`? Anote sus observaciones.
Nota `M0`=$FFFF.

Ejercicio 9.14 Postincremento por 1
Ensamble y ejecute
```
asm p:$200 move a1,X:(R0)+
```
¿Cuál es el contenido de `R0`?, ¿en qué dirección de memoria de X se escribió el contenido de `a1`? Anote sus observaciones.

Ejercicio 9.15 Postdecremento por 1
Ensamble y ejecute

```
asm p:$200 move a1,X:(R0)-
```
¿En qué dirección de memoria X se escribió el dato contenido en `a1`?, ¿cuál es el valor de `R0`? Anote sus observaciones.

Ejercicio 9.16 Postincremento con compensación
Cargue `N0` con $0004. Ensamble y ejecute
```
asm p:$200 move a1,X:(R0)+N0
```
¿En qué dirección de memoria X se escribió el dato contenido en `a1`?, ¿cuál es el valor de `R0`? Anote sus observaciones.

Ejercicio 9.17 Postdecremento con compensación
Se supone que N0 = $0004, por ejemplo anterior. Ensamble y ejecute
```
asm p:$200 move a1,X:(R0)-N0
```
¿En qué dirección de memoria X se escribió el dato contenido en `a1`?, ¿cuál es el valor de `R0`? Anote sus observaciones.

Ejercicio 9.18 Predecremento por 1
Ensamble y ejecute
```
asm p:$200 move a1,X:-(R0)
```
¿En qué dirección de memoria X se escribió el dato contenido en `a1`?, ¿cuál es el valor de `R0`? Anote sus observaciones.

9.1.3. Direccionamiento con los registros `Mn`

En realidad, en los ejemplos anteriores, cada instrucción tomó en cuenta el valor que tenía el registro `M0`, ya que los registros `Mn` alteran la dirección efectiva que se deriva de los registros `Rn` y `Nn`. En la tabla 9.2 se muestran los posibles valores de los registros `Mn` y su efecto en la aritmética de direccionamiento. En resumen se tienen tres posibilidades: `Mn = 0`, `Mn=-1`, y `Mn`$\neq$0 y 1.

El direccionamiento módulo `L` es un modo de direccionamiento muy importante en algoritmos de PDS, donde `L` indica la longitud de la lista de datos. Este modo tiene aplicaciones en la construcción de registros FIFO (*first in first out*), colas circulares, líneas de retardo, registros de (re)corrimiento, sumas finitas de productos, entre otras.

9.1.4. Ajuste del registro `Mn` en módulo `L`

El registro `Mn` trabaja como se indica en la tabla 9.3.

9.1.5. Modo de direccionamiento con inversión de bit

Si `Mn` = $00000, entonces se tiene un direccionamiento que aplica a la inversión de bits para calcular la dirección efectiva. Este modo es particularmente importante en los algoritmos de la transformada rápida de Fourier.

Tabla 9.2: Valores de los Registros `Mn`.

Valor de `Mn`	Aritmética usada
0	Inversión de bits
1	Módulo 2
2	Módulo 3
⋮	⋮
⋮	Módulo L (L = `Mn`+1)
32766	Módulo 32677
32767	Módulo 32768
⋮	Reservado (No se usa)
65535	Lineal (Módulo 65536)

Tabla 9.3: Direccionamiento en módulo `L`.

Registro modificador `Mn` = L-1	Módulo = L
Lím. Inf.=$\overset{15}{X}$X... XX0...$\overset{0}{0}$, (J ceros) donde $2^J \geq L$	Inicio de Tabla†
Lím. Sup.=$\overset{15}{X}$X... XX0...$\overset{0}{0}$=L-1, (J ceros)	Fin de Tabla
Lím. Inf.≤ `Rn`≤ Lím. Sup.	Punto de inicio
Registro de compensación `Nn` = incremento ≤ L	Incremento deseado‡

Notas:
†(minimizar J)
‡(si lo hay).

Tabla 9.4: Direccionamiento con inversión de bits

Registro modificador `Mn`=$0000	N = tamaño de la tabla
Lím. Inf.=$\overset{15}{X}$X... XX0...$\overset{0}{0}$, (K ceros) donde $2^K \geq N$	Inicio de tabla†
Lím. Sup.=$\overset{15}{X}$X... XX0...$\overset{0}{0}$ $+2^K - 1$,	Fin de Tabla
Lím. Inf.≤ Rn≤ Lím. Sup.	Punto de inicio
Registro de compensación `Nn`=2^{K-1}	Inicializa Registro `Nn`

Notas:
†(minimizar K)

En la tabla 9.4 se muestran los ajustes resultantes para este modo de direccionamiento.

Ejercicio 9.19 Direccionamiento con inversión de bits
Sea N=1024, K=10, el límite inferior es 3×N = 3072, el límite superior es 3×N+2^K-1=3072+1024-1=4095, entonces se emplea `R0`=3072 dirección de inicio, `N0`=512 incremento, `M0`=0 elige el modo de direccionamiento con inversión de bits. Siga el procedimiento siguiente:

- Cargue los valores indicados para `R0` y `N0` usando el mando `CHANGE`.
- Luego ensamble la instrucción en `p:$200`
  ```
  asm p:$200 move x:(R0)+N0,x0
  ```
- Ahora ejecute la instrucción con la opción `step`. Se obtiene un nuevo valor para `R0`, anote este resultado.
- Repita la instrucción anterior y anote nuevamente el resultado.
- Finalmente vuelva a ejecutar la instrucción y anote el resultado.
- Observe los últimos tres valores consecutivos de `R0`. Explique y compruebe por qué se dan esos valores.

Ejercicio 9.20 Alternativamente, repita el ejercicio anterior usando una FFT de 8 puntos N=8.

- Modifique los registros con los siguientes valores: `R0=$0000` dirección de inicio, `N0`=4 incremento, `M0`=0.
- Cambie el `pc` a la dirección $200 (recuérdese que en esa dirección está ensamblada la instrucción).
- Ejecute un `STEP` y observe y anote el valor del registro `R0`.
- Cambie el `pc` a $200 y ejecute un `STEP`. Observe y anote nuevamente el valor de `R0`.
- Repita el paso anterior seis veces, observando y anotando en cada caso el valor de `R0`

Al terminar se puede observar que la dirección a la que apunta `R0` es efectivamente la dirección que resulta después de la inversión de bits.

9.2. El juego de instrucciones del DSP56002

El DSP56002 tiene un total de 62 instrucciones que se pueden agrupar en 6 grupos:

- Instrucciones MOV.
- Instrucciones Aritméticas.
- Instrucciones Lógicas.
- Instrucciones de manipulación de bits.
- Instrucciones de control de programa.
- Instrucciones de lazo.

Algunas instrucciones, 30, permiten movimientos en paralelo, lo cual es una característica deseable en un DSP. Por ejemplo, la instrucción:
`MACR -x1,y1,a x:(R0)+,x1 Y:(r4)-,y1`
realiza las siguientes operaciones sobre los datos:

- `MACR` especifica las operaciones de la `ALU`, operación lógica y redondeo convergente.
- Soporta el registro `CC`, código de condición, bits 0 al 5. Multiplica los datos contenidos en `x1`, `y1` y el resultado lo suma al contenido de `a`, además efectúa una operación de redondeo
- Los otros dos campos a la derecha del acumulador `a`, especifican
  - dos transferencias de datos
  - dos diferentes modos de direccionamiento
  - puede trabajar con los espacios de memoria X, Y, L, P, XP.

  Además soporta operaciones de escalamiento, limitación, extensión de signo y relleno de ceros de la parte menos significativa, fuentes duplicadas, pero no soporta destinos duplicados.

Una lista detallada de las operaciones contenidas en esa línea de código es:

1. `x0` es negado.
2. Multiplicación de 24×24 bits, `x1`×`y1`.
3. Suma de 56 bits al acumulador A.
4. Redondeo convergente de A.

5. Actualiza el registro CC.
6. Transfiere el contenido (anterior) del acumulador A al espacio de memoria X.
7. Realiza escalamiento.
8. Realiza limitación si es necesario.
9. Calcula la siguiente dirección `(R0)+N0`.
10. Emplea el registro modificador `M0` para
    - direccionamiento lineal,
    - direccionamiento módulo M, o
    - direccionamiento con inversión de bit.
11. Transfiere el contenido de la memoria Y, de una dirección apuntada por `R4`, al registro `y1`. Si el destino fuera el acumulador B, incluiría extensión de signo y relleno de ceros.
12. Calcula la siguiente dirección `(R4)-N4`.
13. Emplea el registro `M4` para
    - direccionamiento lineal,
    - direccionamiento módulo M, o
    - direccionamiento con inversión de bit.

9.3. Transferencias en paralelo

Los tipos de transferencias en paralelo son:

- Sin transferencia en paralelo.
- Transferencia de dato corto inmediato.
- Transferencia de datos entre registros.
- Actualización de registro de direcciones.
- Transferencia de dato de/hacia memoria X.
- Transferencia de dato entre Memoria XY y registro.
- Transferencia de dato de/hacia memoria Y.
- Transferencia de dato entre registro y memoria Y.
- Transferencia de dato de/hacia memoria L.
- Transferencia de dato de/hacia memoria XY.

9.3.1. Reglas de transferencias en paralelo

Se tienen cinco reglas generales para transferencias en paralelo:

- Sí se permite duplicar la fuente:
  `A` ⟶ `Mem X` y `A` ⟶ `Mem Y`
- No se permite duplicar el destino.
- Si la fuente es un acumulador, entonces puede ocurrir (re)corrimiento o limitación.
- Si el destino es un acumulador, entonces ocurre extensión de signo y relleno de ceros de la parte menos significativa.
- Debido a la estructura *pipeline*, si se modifica un registro de direcciones usando una instrucción `MOVE`, el nuevo contenido del registro no estará disponible para usarlo en la siguiente instrucción. Si `Rn` se modifica con un post o pre-decremento, sí estará listo para la próxima instrucción.

Ejercicio 9.21 Transferencia en paralelo de dato corto inmediato (entero sin signo)
Haga A=$01000001000001, y B=$06FFFFFEFFFFFE y ensamble la siguiente instrucción:

```
asm p:$400 add A,B #$25,A0
```

Muestre los contenidos en A y B antes y después de la ejecución.

Ejercicio 9.22 Transferencia en paralelo de dato corto inmediato (fracción con signo)
Haga A=$01000001000001, y B=$06FFFFFEFFFFFE y ensamble la siguiente instrucción:

```
asm p:$400 add B,A #$C7,B
```

Muestre los contenidos en A y B antes y después de la ejecución.

Ejercicio 9.23 Transferencia en paralelo de Registro a Registro
Haga A=$01000001000001, y B=$12654321FEDCBA, N7=$0000 y ensamble la siguiente instrucción:

```
asm p:$400 add A,B A,N7
```

Muestre los contenidos en A, B y en N7, antes y después de la ejecución. La transferencia en paralelo es `A,N7`.

Ejercicio 9.24 Actualización en paralelo de registro de direcciones
Haga A=$01000001000001, y B=$06FFFFFEFFFFFE, R1=$2000, N1=$F000

y ensamble la siguiente instrucción:

```
asm p:$400 add A,B (R1)+N1
```

Muestre los contenidos en A, B, `R1`, y `N1`, antes y después de la ejecución.

Ejercicio 9.25 Actualización en paralelo de registro de direcciones
Haga A=$01000001000001, y B=$12123456123456, R1=$1000, N1=$0000 y ensamble la siguiente instrucción:

```
asm p:$400 add A,B (R1)-
```

Muestre los contenidos en A, B, y `R1` antes y después de la ejecución.

Ejercicio 9.26 Transferencia paralelo de/hacia memoria de datos X
Haga A=$00123456ABCDEF, y B=$00123456123456, X:$1000=$0002, y1=$000123 y ensamble la siguiente instrucción:

```
asm p:$400 add y1,B A,X:$1000
```

Muestre los contenidos en X, B, `y1` antes y después de la ejecución.

Ejercicio 9.27 Transferencia paralelo de/hacia memoria de datos X
Haga A=$00123456ABCDEF, y B=$00123456123456, X:$1000=$0002, y1=$000123, R6=$1001 y ensamble la siguiente instrucción:

```
asm p:$400 add  y1,B X:-(R6),A
```

Muestre los contenidos en X, B, `R6` y A antes y después de la ejecución.

Ejercicio 9.28 Transferencia paralelo de memoria X y registro
Haga A=$00222222333333, y1=$000000 X=$222222222222, X:$1005=$0002, N3=$0005, R3=$1000 y ensamble la siguiente instrucción:

```
asm p:$400 add X,A A,X:(R3+N3) A,y1
```

Muestre los contenidos de la memoria X y los registros B, `R6 y1`, y A, antes y después de la ejecución.

Ejercicio 9.29 Transferencia paralelo de memoria Y
Haga A=$00222222333333, y0=$000000 X=$222222222222, X:$1000=$0002, N3=$0005, R3=$1000 y ensamble la siguiente instrucción:

```
asm p:$400 add y0,A A,x1 A,Y:(R3)-N3
```

Muestre los contenidos de la memoria Y y los registros `R3 y1`, `x1` y A, antes y después de la ejecución.

Ejercicio 9.30 Transferencia paralelo de memoria larga (L)
Haga B=$00222222333333, y=$505050404040, X:$2000=$222222, Y:$2000=$111111 N3=$0005, R3=$2000 y ensamble la siguiente instrucción:

```
asm p:$400 add Y,B BA,L:(R3+N3) A,y1
```

Muestre los contenidos de los registros mem-X, y mem-Y, B, R3, Y, antes y después de la ejecución.

Ejercicio 9.31 Transferencia paralelo de memoria XY
Haga B=$00222222333333, y=$505050404040, x=$111111111111, X:$2000=$222222, Y:$1000=$111111, R1=$2000 R5=$1000 y ensamble la siguiente instrucción:

```
asm p:$400 add X,B X:(R3)+,x1 Y:(r5)-,y0
```

Muestre los contenidos de los registros mem-X, y mem-Y, B, R3, R5, X y Y, antes y después de la ejecución.

Existe también la transferencia de registros de control, de memoria P, de datos de los periféricos, así como instrucciones de manipulación de bits, operaciones lógicas, entre otras. Para una lista completa de instrucciones vea el manual de usuario.

9.4. La instrucción REP y la instrucción DO

La instrucción REP repite la siguiente instrucción el número de veces indicado en un registro o en modo inmediato.

9.4.1. Características de la instrucción REP

- Sólo son repetibles instrucciones de una sola palabra.
- La instrucción REP y la siguiente son capturadas una sola vez y almacenadas internamente.
- REP no se puede interrumpir.
- Si LC = 0 especifica el máximo número de repeticiones de 65,536.
- REP no puede ser la última instrucción de un lazo DO.
- Las siguientes instrucciones no son repetibles:
  - DO
  - Jcc
  - JCLR
  - JMP

- JSET
- JScc
- JSCLR
- JSR
- JSSET
- REP
- RTI
- RTS
- STOP
- SWI
- WAIT

Ejercicio 9.32 La instrucción REP
Defina x0=$000005, A=$00222222333333, R0=$1000, R4=$2000. Ensamble la siguiente instrucción:

```
asm p:$100 rep x0
asm p:$101 move A,X:(R0)+
```

No olvide cambiar el pc a $100. Asegúrese de abrir una ventana de datos X:$1000. Finalmente, ejecute STEP 5, o bien use el botón STEP cinco veces. Revise el contenido de la memoria X:$1000 a X:$1004

9.4.2. La instrucción DO

La instrucción DO, a diferencia de la instrucción REP, repite un bloque de instrucciones especificando en la línea de la instrucción DO, con una etiqueta, el final del código a repetir.

La sintaxis es la siguiente:

```
      DO    y0,FIN
      .
      .
      .
      MAC  x1,y1,A  X:(R0)+,x1  Y:(R4)+,y1
      .
      .
FIN   ADD  x1,A  ;Primera instrucción después del lazo
```

FIN es una etiqueta para indicar el fin del lazo, y y0 contiene el número de veces que se repetirá la instrucción DO.

Bibliografía

[1] Freescale Semiconductor, Inc., *DSP56000/56001 Digital Signal Processor User's Manual,*
DSP56000UM/AD Rev 2. Motorola Inc. 1990.

[2] Freescale Semiconductor Inc., *DSP56002 Digital Signal Processor User's Manual,*
DSP56002UM/AD Rev 1. Motorola Inc. 1993.

[3] Course Notes MTT31/CN Rev. 3.0, 1990, Freescale Semiconductor, Inc.